A prototype for the Real-Time Analysis of the Cherenkov Telescope Array

Andrea Bulgarelli*[a], Valentina Fioretti[a], Andrea Zoli[a], Alessio Aboudan[a,b], Juan José Rodríguez-Vázquez[c], Gernot Maier[d], Etienne Lyard[e], Denis Bastieri[f,k], Saverio Lombardi[k], Gino Tosti[g], Adriano De Rosa[a], Sonia Bergamaschi[a,h], Matteo Interlandi[a.h], Domenico Beneventano[a,h], Giovanni Lamanna[i], Jean Jacquemier[i], Karl Kosack[j], Lucio Angelo Antonelli[k], Catherine Boisson[l], Jerzy Borkowski[m], Sara Buson[f], Alessandro Carosi[k], Vito Conforti[a], Jose Luis Contreras[n], Giovanni De Cesare[a], Raquel de los Reyes[o], Jon Dumm[p], Phil Evans[q], Lucy Fortson[p], Matthias Fuessling[s], Ricardo Graciani[r], Fulvio Gianotti[a], Paola Grandi[a], Jim Hinton[q], Brian Humensky[t], Jürgen Knödlseder[u], , Giuseppe Malaguti[a], Martino Marisaldi[a], Nadine Neyroud[i], Luciano Nicastro[a], Stefan Ohm[q], Julian Osborne[q], Simon Rosen[q], Alessandro Tacchini[a], Eleonora Torresi[a], Vincenzo Testa[k], Massimo Trifoglio[a], Amanda Weinstein[v] for the CTA Consortium[w]

[a] INAF/IASF Bologna, Via Gobetti 101, 40129 Bologna (Italy)
[b] CISAS, University of Padua, Padua, Italy
[c] Centro de Investigaciones Energéticas, Medioambientales y Tecnologicas, Madrid, Spain
[d] Deutsches Elektronen-Synchrotron, Platanenallee 6, Zeuthen, Germany
[e] University of Geneva, Switzerland ISDC Data Center for Astrophysics, Chemin d'Ecogia 16, 1290 Versoix
[f] Dipartimento di Fisica, Università degli Studi di Padova, Italy
[g] Università di Perugia, Italy
[h] University of Modena and Reggio Emilia, Department of Engineering "Enzo Ferrari", Strada Vignolese 905 - 41125 Modena (Italy)
[i] Laboratoire d'Annecy-le-Vieux de Physique des Particules, Université de Savoie, CNRS/IN2P3, France
[j] CEA/DSM/IRFU, Saclay, France
[k] INAF - Osservatorio Astronomico di Roma, Italy
[l] Observatoire de Paris, LUTH, CNRS, Universite´ Paris Diderot, France
[m] Copernicus Astronomical Center, Polish Academy of Sciences, Poland
[n] Dept. FAMN, Universidad Complutense de Madrid, Grupo de Altas Energías, Spain
[o] Max-Planck-Institut für Kernphysik, P.O. Box 103980, D 69029 Heidelberg, Germany
[p] University of Iowa, Department of Physics and Astronomy, USA
[q] Dept. of Physics and Astronomy, University of Leicester, United Kingdom
[r] Departament d'Astronomia i Meteorologia, Institut de Cie`ncies del Cosmos, Universitat de Barcelona, Spain
[s] Institut fr Physik und Astronomie, Universität Potsdam, Karl-Liebknecht-Strasse 24/25, D 14476 Potsdam, Germany
[t] Department of Physics and Astronomy, Barnard College; Department of Physics, Columbia University, USA
[u] CNRS/IRAP; 9 Av. colonel Roche, BP 44346, F-31028 Toulouse cedex 4, France
[v] Department of Physics and Astronomy, Iowa State University, Ames, IA 50011, USA
[w] see http://www.cta-observatory.org/ for full author and affiliation list

Ground-based and Airborne Telescopes V, edited by Larry M. Stepp, Roberto Gilmozzi, Helen J. Hall, Proc. of SPIE Vol. 9145, 91452X • © 2014 SPIE
CCC code: 0277-786X/14/$18 • doi: 10.1117/12.2054744

ABSTRACT

The Cherenkov Telescope Array (CTA) observatory will be one of the biggest ground-based very-high-energy (VHE) γ-ray observatory. CTA will achieve a factor of 10 improvement in sensitivity from some tens of GeV to beyond 100 TeV with respect to existing telescopes.

The CTA observatory will be capable of issuing alerts on variable and transient sources to maximize the scientific return. To capture these phenomena during their evolution and for effective communication to the astrophysical community, speed is crucial. This requires a system with a reliable automated trigger that can issue alerts immediately upon detection of γ-ray flares. This will be accomplished by means of a Real-Time Analysis (RTA) pipeline, a key system of the CTA observatory. The latency and sensitivity requirements of the alarm system impose a challenge because of the anticipated large data rate, between 0.5 and 8 GB/s. As a consequence, substantial efforts toward the optimization of high-throughput computing service are envisioned.

For these reasons our working group has started the development of a prototype of the Real-Time Analysis pipeline. The main goals of this prototype are to test: (i) a set of frameworks and design patterns useful for the inter-process communication between software processes running on memory; (ii) the sustainability of the foreseen CTA data rate in terms of data throughput with different hardware (e.g. accelerators) and software configurations, (iii) the reuse of non-real-time algorithms or how much we need to simplify algorithms to be compliant with CTA requirements, (iv) interface issues between the different CTA systems. In this work we focus on goals (i) and (ii).

Keywords: Cherenkov Telescope Array, real-time Analysis, gamma-ray flares, Cherenkov telescopes, VHE gamma-ray astronomy, science alert system, GPU, Intel/Phi

1. INTRODUCTION

The Cherenkov Telescope Array (CTA)[1, 2] will be one of the biggest ground-based very-high-energy (VHE) γ-ray observatory an may be built by the end of this decade. In the preparatory phase (2011-2014) at the time of writing, the international CTA consortium counts more than 1000 scientists from 28 countries.
CTA will consist of two arrays: one in the southern hemisphere, to observe the wealth of sources in the central region of our Galaxy, and one in the north, primarily devoted to the study of Active Galactic Nuclei (AGN)[3], galaxies at cosmological distances, and star formation and evolution. To accomplish the science goals, tens of telescopes (Figure 1) of three different sizes will be required: a Large Size Telescope for the lowest energies (20 GeV - 1 TeV), a Medium Size Telescope for the 100 GeV - 10 TeV energy domain, and the Small Size Telescope for the highest energies (few TeV - beyond 100 TeV). Thanks to the large number of individual telescopes, CTA can operate in a wide range of configurations, enabling operations with multiple sub-array targeting and simultaneous monitoring of different objects or energy ranges.

Among many outstanding features, CTA will achieve a factor of 10 improvement in sensitivity from some tens of GeV to beyond 100 TeV with respect to existing Cherenkov observatories: HESS[4], MAGIC[5] and VERITAS[6]. For this reason the observatory will have a large discovery potential in astrophysics and fundamental physics and will try to shed light on the inception of cosmic rays, the nature of black holes and their role as particle accelerators and the physics of matter beyond the Standard Model (dark matter and quantum gravity).

To maximize the science return on time-variable and transient phenomena, the CTA Observatory must be capable of issuing alerts of transient events from astrophysical sources changing the target (i.e. transition from data-taking on one target to data-taking on another target) anywhere in the observable sky. The main purpose of these science alerts is to re-schedule the observations to follow the phenomena and to issue alerts to other facilities. With its large detection area, CTA can resolve flaring and time-variable emission on sub-minute time scales.

In order to issue alerts when time-variable and transient phenomena are detected a Real-Time Analysis (RTA) system is currently being designed. A Science Alert System that will generate scientific alerts is also one of the key systems of the CTA observatory.

2. THE CTA REAL-TIME ANALYSIS

The RTA is expected to generate science alerts with a latency of 30 s starting from the last acquired event that contributes to the alert. The sensitivity of the analysis is required to not be worse than the one of the final analysis by more than a factor of 3, with an availability of the RTA during observations of greater than 98%. In addition, the search for transient phenomena must be performed on multiple timescales (i.e. using different integration time windows) from seconds to hours, both within a defined source region, and elsewhere in the field of view. Last but not least, there will be a RTA pipeline for each sub-array running in parallel.

These requirements shall be fulfilled taking into account some constraints that the peculiarity of the Observatory imposes. CTA is expected to generate a large data rate, due to the large number of telescopes, the amount of pixels in each telescope camera and the fact that several time samples are recorded in each pixel of the triggered telescopes. Depending on (i) the **array layout**, (ii) the **data reduction schema** adopted, and (iii) the **trigger criteria**, the data rate estimations vary from 0.5 to 8 GiB/s. This high data rate coupled with the location of the two arrays, where it is reasonable to suppose that the candidate sites will have a limited bandwidth for data transfer, imply that the RTA will be performed on-site as an essential component of the CTA On-Site Analysis, on hardware located at the telescopes site.

The RTA is part of the CTA On-site Analysis infrastructure[7].

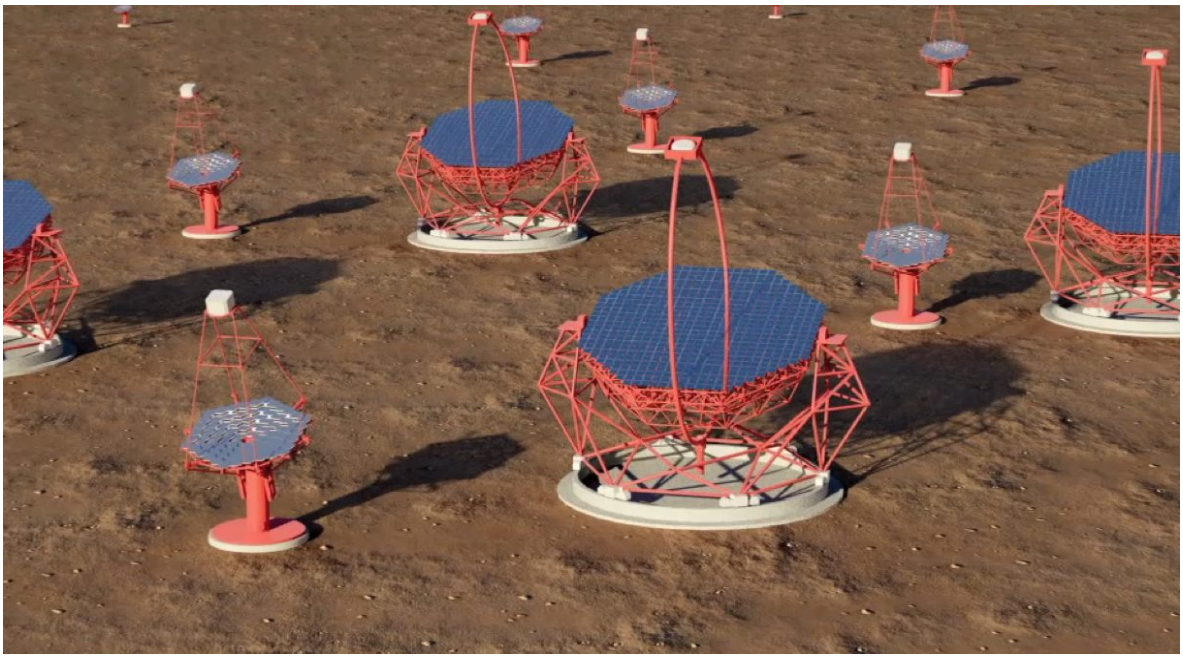

Figure 1: An artist impression of the Cherenkov Telescope Array. Source G. Perez, Instituto de Astrofísica de Canarias, Servicio Multimedia (SMM/IAC).

The RTA interfaces with the following external functionalities:

1. DAQ (Data Acquisition System), that acquires the triggered and calibration data from the cameras and builds them into a single Cherenkov event* using an Event Builder†;
2. Short-term scheduler, that receives the Science Alert generated by RTA;
3. Monitoring system, that provides health and performance monitoring of the overall CTA system.

2.1 The design of the Real-Time Analysis

The RTA processes the data acquired by the DAQ during the telescopes data acquisition. The data are analyzed using standard or blind search methods to detect known or serendipitous astrophysical sources in a high or unexpected state, select the best transient candidates and generate Science Alerts.

To fulfill the requirements of the RTA, taking into account the constraints in terms of data rate we have designed the RTA as an **on-line system** (the data are analyzed during the data acquisition)‡ and an **on-site system** that acquires data from the DAQ system of the CTA. Some alternative schema are foreseen, e.g. it will be possible to use some components of the RTA within the DAQ for data reduction purposes, using some steps of the RTA pipeline as a pre-storage analysis for event pre-selection.

On-the-fly **data streaming** is one of the basic requirements for an on-line system in general, and for this system in particular; this means that a **stream of bytes** is transmitted between the different processes and formatted as a logical sequence of bits, which represent well-defined information.

The data streaming from the DAQ enables us to reduce the latency between the data acquisition and the data processing because no data§ or temporary results are written to disk: the data acquired by the DAQ is sent directly to the first component of the RTA. The stream of data coming from the DAQ contains the data of an *event* or of a *triggered camera event*** of a CTA telescope.

Also in the subsequent steps of the RTA pipeline the concept of the data streaming is maintained. This data streaming enables the **stream processing** of the incoming data: the data are analyzed continuously as soon as they are acquired, reducing to the minimum level, the latency of the entire system.

Due to the foreseen data rate, coupled with the required latency for alert generation, the RTA will run a **simplified analysis** of data running algorithms optimized for real-time purpose.

In the final configuration there will be **multiple pipelines**, one for each sub-array.

3. THE PROTOTYPE

To understand how to fulfill the CTA *requirements* and to help the definition of the RTA *specifications* we have developed a prototype with the main aim of proving the feasibility of the Real-Time Analysis of CTA, that imposes very dramatic constraints in terms of response time and sensitivity with a very high acquisition data rate.

---

* A Cherenkov event, or **event**, in the CTA nomenclature, is a transient (<1 μs) illumination of one or more of the CTA telescopes with UV-optical light, usually due to Cherenkov emission from an atmospheric cascade initiated by a cosmic photon, electron, proton or nucleus, that is registered by the CTA system.

† A component that assembles the triggered camera data all together with a unique event identifier.

‡ In contrast with the **off-line analysis**, where the analysis is performed after the data acquisition, typically at the end of an observation or the next morning.

§ The data acquired by the DAQ will be stored on-disk by the DAQ system or by the Preliminary Analysis pipeline, and off-line and on-site system.

** A **triggered camera event** is a transient illumination of a camera of a CTA telescope due to Cherenkov light. The **camera event data** are the complete information (registered signal and their time-stamp from all the pixels) of the events triggered by an Array trigger, merged in a single data structure. The content of the data depends on the data taking scenario, and spans from the acquisition of the pulse waveform for all the pixels to the integral digitized signal for each pixel and event. In addition, different zero-suppression schemas may be applied.

The main purposes of the prototype are:

1) to prove the the feasibility of the *stream processing* performed on CTA data;
2) to test a fast inter-process communication architecture: in this version of the prototype we have focused on processes running on the same computing node (in-memory data transfer);
3) to test the data transfer between CPU and GPUs (Graphics Processing Unit) and the feasibility of the *stream processing* with these hardware accelerators. The current on-line and off-line software of current Cherenkov experiments[8, 9, 10, 11] and the AGILE space mission on-line analysis[12] do not use GPUs; the Fermi team has developed a prototype of its off-line analysis†† using GPU.

For the development of this technological assessment (see Figure 2), first we developed a simulator of the CTA Event Builder (*EBSim*) using as input the CTA Monte Carlo (MC) *"PROD2"* data[16]. The MC data are converted into a raw data format, a stream of bytes that are transmitted between the different processes of the RTA pipeline. *EBSim* pre-loads the MC data into a circular buffer and generates a data stream of 0.1-3 GB/s simulating the expected CTA data flow.

We have also defined a first software architecture able to perform some steps of the Real-Time Analysis pipeline. A set of libraries manages the data format and the metadata. The Packetlib and RTAtelem software libraries decode and route the data; RTAConfig provides the configuration of the array and of the cameras; libQLBase provides some basic functionalities (e.g. file access). See the following subsections for more details.

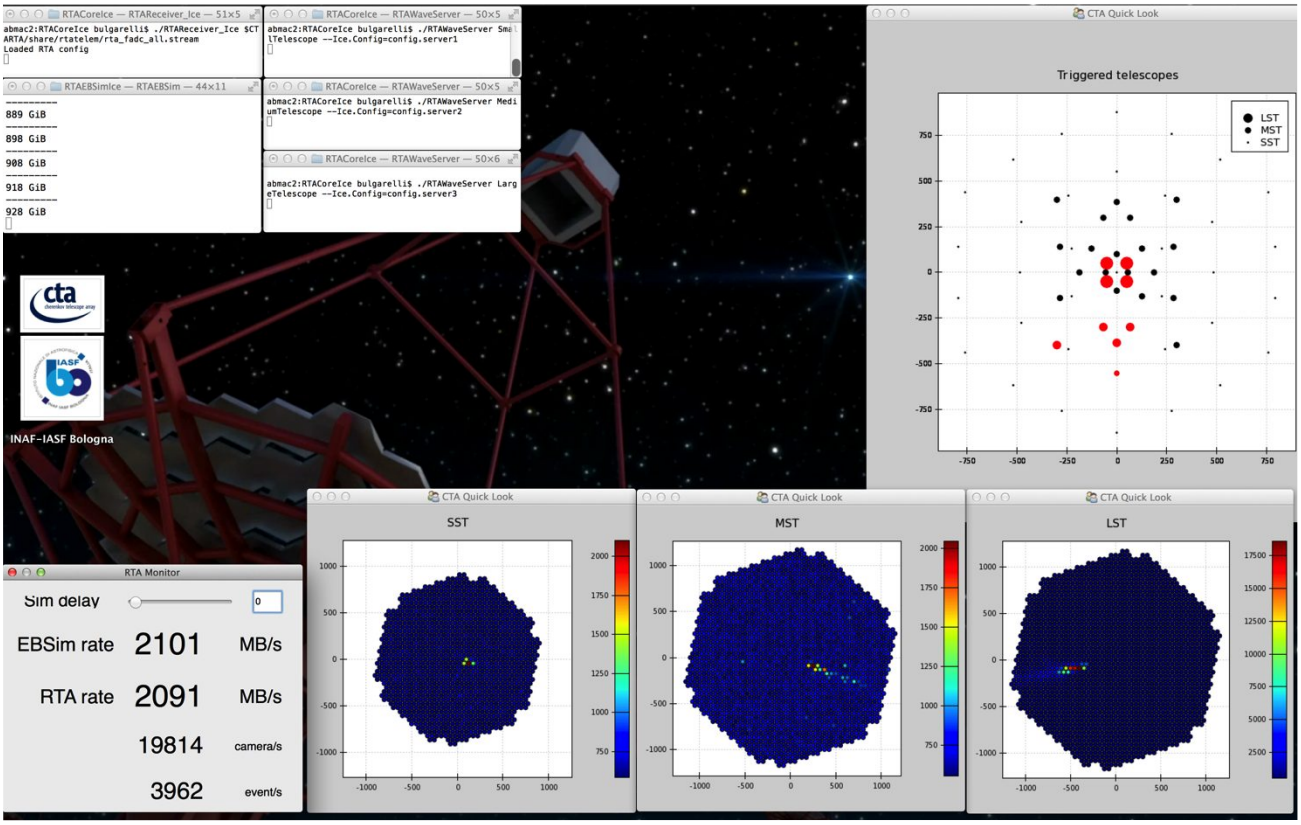

Figure 2: The RTA prototype in action. The Event Builder simulator sends the camera data to the RTAReceiver process, that decode the packet and sends them to different RTAWaveServers. Each RTAWaveServer (one for each type of telescope: SST, MST, LST, with a different number of threads for load balancing) performs a waveform extraction algorithm (as described in the text) and sends the result to some graphical displays of the cameras (bottom/right). The RTAReceiver identifies a Cherenkov event in the stream of data and sends the information of the event to an Array visualizer (top/right, red circles indicates the triggered telescopes). A RTA Monitor displays the data rate and event rate (events/s) sustained by this prototype.

†† http://www.nvidia.com/content/cuda/spotlights/gpu-accelerated-astronomy.html

The *RTACore* is the entry point of the RTA prototype and acquires and routes the data to the correct "telescope type process" for data reduction (the *RTAWaveServer* components). A subset of the events is sent also to an Array Viewer to perform a fast quick look of the events in real-time (see Figure 2).

The *RTAWaveServer* performs a waveform extraction algorithm (see the following section) on a CPU. A sample of the reduced events is sent to a Camera Viewer (there is one viewer for each type of telescope) that shows the reduced data (see Figure 2). The integration of these nodes with a GPU version of the waveform extraction algorithm[13] is in progress.

An *RTAMonitor* checks the performance of the pipeline and it is also used to set a delay in the data flow generation; this is useful to check the bottlenecks of different software processes.

ZeroC/Ice[‡‡] is used as the framework for inter-process communication (IPC), using C++ and Python programming languages. The greater part of the developed code is framework independent and, for this reason, we have separated the management of the data (performed by ad hoc software libraries) with the IPC mechanisms (provided by Zeroc/Ice): this approach allows a maximization of the portability of the RTA software to different or multiple software frameworks.

3.1 PacketLib and RTATelem

One of the main building blocks of the RTA prototype presented in this work is the PacketLib[14], an open-source C++ software library (released under GPL license)[§§] that optimizes the streaming data processing for a real-time analysis system. The library encodes, decodes and routes data in a structured data format (basically a stream of formatted bytes) organized in **packets**, a chunk of data with one or more headers and a data section; each section is a set of **fields** (the atomic unit of information contained into a packet, basically a set of bits) and the header must contains, at least, the packet length (the size of the packet in bytes) and one or more fields used to identify the structure of the packet. PacketLib could manage also satellite telemetry source packets (compliant with CCSDS/ECSS ESA standard[22]) and in this context has been used in the development of the Ground Segment and of the Test Equipment of some space missions: ASI/AGILE[12], CNES/COROT, and ESA/Bepi Colombo Serena. In the ground-based telescope context PacketLib is being used in the DAQ of the ASTRI Prototype[15]. We have used PacketLib also on embedded systems.

In the RTA prototype context a *packet* is a single event of a camera (e.g. all the pixels of a camera, or all samples of all pixels of a camera, depending on hardware and readout schema). The data of a single triggered camera is the basic chunk of data that this prototype should receive. This choice has been made to gain flexibility because, at the time of writing, the decision if RTA should receive single camera events or a set of cameras events representing a Cherenkov event has not been decided yet.

From an input stream of bytes (it does not matter if this stream has been generated with this library or not), the library is able to recognize automatically the packets (described by configuration files), and provides a simple access to each packet field by means of an object-oriented interface (see Figure 3). In order to improve the performances of the stream processing, the data field decoding, which is a time consuming operation, is performed on demand. This choice greatly increase the performances, in particular for the following operations:

- *data routing*, to send each type of packet to different processes (data analysis, storage, etc). To perform the data routing it is necessary to decode only the sections of the packet that contains the fields of the identifier. Thanks to this identifier we can recognize the entire content of a packet by decoding only a few bits without decoding the rest of the packet;
- *data access* to a block of data (e.g. all samples of a pixel, all pixels of a camera) within the packet and move them to a hardware accelerator. To speed-up this operation PacketLib identifies only the starting address of the required sections without decoding fields.

[‡‡] http://www.zeroc.com
[§§] http://www.gnu.org/copyleft/gpl.html

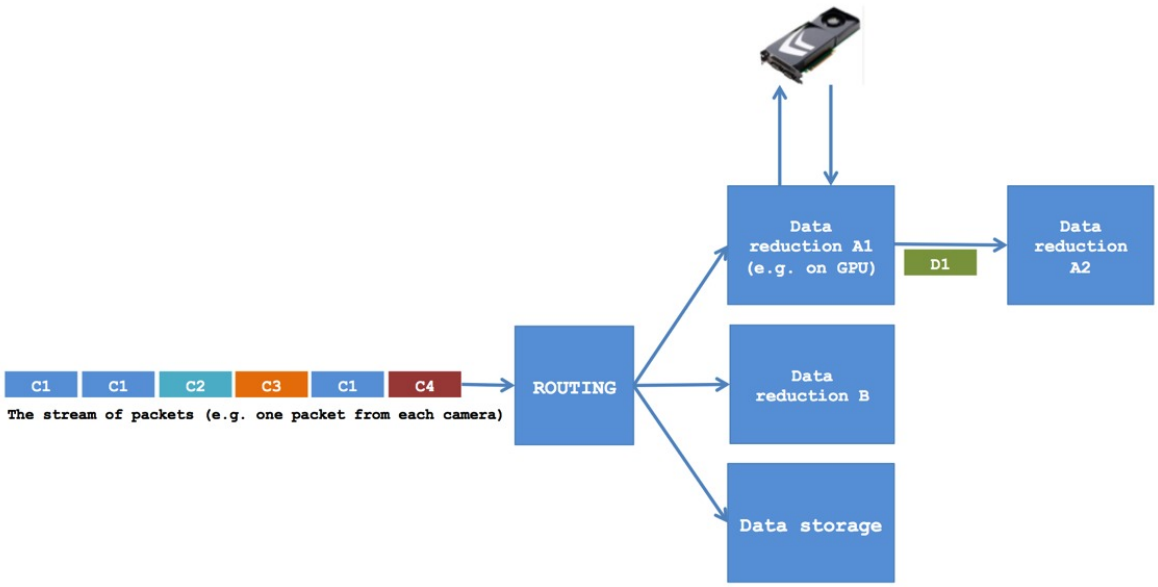

Figure 3: the streaming of packets (triggered camera data) and some operations performed by the PacketLib: routing, decoding at block level to move the data to CPU or GPU, generation of a new packet type

In addition, PacketLib allows generations of new packets from the result of data reduction algorithms (see D1 in the Figure 3) and sends them in streaming to another process, or stores them in the archive. In case a cyclic redundancy check (CRC) field is present within the packet, PacketLib can be used to perform an on-the-fly data-quality check on each single packet to verify data corruption during the *stream processing*.

The *data compression* is an important factor to reduce the data rate. PacketLib is able to perform a data compression/ decompression on-the-fly at the level of single packet. To manage compressed packets in streaming PacketLib compresses only the data section, leaving the headers decompressed[***]; this enables reading, storing and routing packets without decompressing them and decompresses the data only when needed (on-demand, e.g. only before the data reduction). With this approach we can increase the data streaming rate depending on the compression algorithm used in the PacketLib, without any change in the proposed RTA architecture. With PacketLib compressed and uncompressed data are handled transparently.

RTATelem is a collection of classes that wrap the PacketLib data format for its use in the RTA pipeline..

3.2 The data input and RTAConfig

The RTA pipeline uses as input the "PROD2" CTA Monte Carlo (MC) simulated data[16], kindly provided by the CTA MC team and based on the array configuration of the Aar (Namibia) proposed south site. In particular, three types of telescopes compose the array: the small (35 telescopes), medium (23 telescopes), and large (4 telescopes) types. The triggered events are stored in a raw form (no calibration is applied) as 16-bit Fast Analog Digital Converter (FADC) count waveforms as a function of the time slice (**sample**) for each camera pixel. In the input simulated data file, the Small Size Telescope (SST) camera is composed by 1141 pixels, and 40 time samples are taken for each pixel. The Medium and Large Size Telescopes (MST and LST) cameras instead present 1855 pixels and 30 samples. The FADC waveform is the core of the triggered event, and its dimension, for each telescope, is given by the number K of pixels multiplied by the number M of samples, $K \times M$.

[***] In a typical packet with pulse waveform containing a camera event the size of the header is of the order of 10 bytes and the size of the data is of the order of 100 KiB: the size of the header is negligible.

The MC standard hessio/eventio format is converted to a ROOT file using the *evndisplay* software package for data analysis developed by the VERITAS collaboration[17]. The camera FADC waveforms are encoded into a binary file using the RTAtelem library. Each triggered camera becomes a packet for streaming data processing.
RTAConfig provides, given the triggered telescope unique identifier, the configuration of the telescopes (e.g. position in the array/sub-array), the telescope type (e.g., SST), its mirror and camera design, up to the pixel position in the camera, which are loaded at run-time.

3.3 The Waveform extraction algorithm

In the development of the prototype it is important to realize a real data taking and data reduction scenario to have a clear understanding of hardware and software capabilities of the proposed solution. To satisfy this, we have developed a sliding window with amplitude-weighted time signal extraction algorithm[18], because (i) it should be the entry stage of the Reconstruction pipeline and, (ii) in spite of its simplicity, is able to exert a lot of pressure on the memory and processing subsystem, and (iii) the data reduction after this processing is of an order of 100 between input and output. This signal extraction algorithm searches for the maximum integral content among all possible windows of fixed size contained in a defined time range.

We have developed many versions of this algorithm both in CPUs (both at IASFBO and at CIEMAT) and on GPUs (at CIEMAT[13]). The integration of the GPU version of this algorithm into this prototype is still in progress. All proposed algorithms work with an array of short-integer samples and with the appropriate arrangement when applicable.

4. RESULTS

In the first version of the prototype everything runs on the same computer (a notebook, an Intel Core i7 2.6 GHz, 16 GB 1600 MHz DDR3 RAM) and the data transfer is performed only in memory: no network data transfer and disk access are tested. We have run the prototype for many hours (simulating a data acquisition of a entire night) and the prototype has proven stable both in terms of memory management and in terms of sustained data rate. We have not registered a degradation of the sustained data rate at the end of a simulated night of observation. Measurements were accomplished through code instrumentation.

The sustained data rate of the first version of the prototype is between 2 and 2.4 GiB/s, with or without array and camera viewers. This is equivalent to an array trigger rate of about 4 kHz with the camera data with full waveforms. The foreseen data rate of the array spans from 7 to 13 kHz[19]. Although the addition of more steps of the reconstruction algorithms will reduce the RTA prototype data rate, a consistent increment of the prototype performance is expected from the following aspects:

1) the current version of the prototype runs on a notebook in a single CPU with a maximum of 8 threads. There is a huge space for the scalability of the system here (adding more CPUs and using professional servers);
2) the producer node (the *Event Builder* simulator) and the RTA prototype share the same computing power (the same CPU): this is not a realistic case, because the RTA pipeline will have a dedicated computer infrastructure.
3) at the time of writing we are not using hardware accelerators;
4) the waveform extraction algorithm is one of the most data throughput intensive algorithms (it is the first step of the RECO pipeline), and the data reduction is of an order of 100 in terms of GiB.

5. CONCLUSIONS

The first version of the RTA prototype presented in this paper is able to sustain an array trigger rates of about 4 kHz running on a single notebook. The current prototype performs data acquisition of array events, data routing between different software processes, a waveform extraction algorithm on a CPU that reduces the data rate by an order of 100, and a graphical display of the array and camera events. At the end all the pipeline components operate in a consistent way maximizing the total system throughput. With this prototype we have proven the feasibility of the *stream processing* performed on CTA data, testing some custom software libraries and a *pipeline* architecture.
The next steps will be to test the inter-process communication via network using all the components of the prototype, to finalize the integration of the GPU version of the waveform extraction algorithm, to test the integration of this prototype

with the CTA DAQ, with the monitoring and archiving system of CTA and with ACS[†††], and to add a real-time version of more algorithms of the reconstruction pipeline.
The introduction of hardware accelerators (GPUs) into the pipeline, the use of more advanced software architectural solutions and the use of more professional Information and Communication Technology (ICT) infrastructure will increase greatly the performance of this prototype.

ACKNOWLEDGMENTS

We gratefully acknowledge support from the agencies and organizations listed under Funding Agencies at this website: http://www.cta-observatory.org/.

REFERENCES

1. Actis, M., et al., "Design concepts for the Cherenkov Telescope Array CTA: an advanced facility for ground-based high-energy gamma-ray astronomy," Experimental Astronomy 32, 193–316 (Dec. 2011)
2. Acharya, B.S., et al., "Introducing the CTA concept", Astroparticle Physics 43, 3–18 (2013)
3. A. Zech et al., Fermi Proceedings eConf C1111101 arXiv:1205.1459v1 (2012)
4. H.E.S.S. Collab.: Aharonian, F. et al., "Observations of the crab nebula with hess," A&A 457, 899 (2006).
5. MAGIC Collab.: Albert, J. et al., "Vhe gamma-ray observation of the crab nebula and its pulsar with the magic telescope," ApJ 674, 1037 (2008).
6. VERITAS Collab.: Acciari, V. A. et al., "Veritas observations of the -ray binary ls i +61 303," ApJ 679, 1427 (2008).
7. Bulgarelli, A., et al., "The Real-Time Analysis of the Cherenkov Telescope Array Observatory", 33rd. International Cosmic Ray Conference, Rio de Janeiro (Brazil), (2013)
8. S. Funk, "Online Analysis of Gamma-ray Sources with H.E.S.S." Ph.D. dissertation, Humboldt University of Berlin, (2005).
9. D. Tescaro, et al., "The MAGIC telescopes DAQ software and on-the-fly online analysis client." 33rd. International Cosmic Ray Conference, Rio de Janeiro (Brazil), (2013).
10. H. Krawczynski, M. Olevitch, G. Sembroski, and K. Gibbs, "Control Software for the VERITAS Cherenkov Telescope System," pp. 2843–2846, (2003).
11. J. S. Perkins, "Control, Monitoring and Analysis Software for the VERITAS Array," AIP Conf. Proc., vol. 921, pp. 564–565, (2007).
12. A. Bulgarelli, et al., "The AGILE Alert System for Gamma-Ray Transients," *The Astrophysical Journal*, vol. 781, no. 1, p. 19, 2014.
13. J. Rodríguez-Vázquez, J.L. Vázquez-Poletti, C. Delgado, A. Bulgarelli and M. Cárdenas-Montes, "Performance evaluation of a signal extraction algorithm for the Cherenkov Telescope Array's Real Time Analysis pipeline", IEEE Cluster Conference, (2014)
14. Bulgarelli, A., Gianotti, F., and Trifoglio, M., "PacketLib: A C++ Library for Scientific Satellite Telemetry Applications," Astronomical Data Analysis Software and Systems XII ASP Conference Series, 295, (2003).
15. Conforti, V., et al., "The ASTRI SST-2M telescope prototype for the Cherenkov Telescope Array. Camera DAQ Software Architecture", these proceedings
16. K. Bernlöhr et al., "Monte Carlo design studies for the Cherenkov Telescope Array", Astroparticle Physics 43 171–188 (2013)
17. G. Meier, "A short description of an evndisplay-based CTA analysis", CTA Internal note
18. J. Albert, et al., "FADC signal reconstruction for the MAGIC telescope," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 594, no. 3, pp. 407 – 419, (2008).
19. M. P. Arribas, et al., "Trigger and data rates expected for the CTA Observatory", arXiv:1211.3061v1
20. Schwarz, J., Farris, A.,Sommer, H.,"The Alma Software Architecture" Proc. SPIE 5496, 190-204 (2004).
21. ECSS secretariat, ECSS-E-70-41A (2003)

[†††] Alma Common Software[20]